Développement d'un pôle de calcul : CESAR-LCPC

Alain DUBOUCHET
Technicien supérieur
Section Modèles numériques
Service Mécanique
Laboratoire Central des Ponts et Chaussées

RÉSUMÉ

CESAR-LCPC, progiciel de calcul par éléments finis, est né au début des années quatre-vingt de la critique des systèmes précédemment développés au LCPC, et de l'étude des méthodes préconisées alors pour organiser de tels outils. Il s'est développé dans les différents domaines du génie civil grâce à une architecture modulaire et à une méthodologie précise, décrites dans un Manuel de programmation. Jusqu'à maintenant, les développeurs avaient retenu pour leur production des critères de qualité inspirés essentiellement par l'expérience acquise. Devenue une exigence chez un nombre croissant d'utilisateurs, la qualité doit maintenant être prise en compte dans le sens formalisé par les travaux de l'AFCIQ et de l'AFNOR. De façon à être présente à tous les stades d'élaboration du progiciel, cette formalisation se doit d'être introduite sous forme de règles et de conseils dans le Manuel de programmation de CESAR-LCPC. S'appuyant sur des critères de qualité clairement affichés, la sous-traitance pourrait alors être envisagée pour les domaines ne constituant pas strictement le « métier » de l'équipe de développement.

MOTS CLÉS : *20 - Programme de calcul - Éléments finis (méthode) - Historique - Développement - Qualité - Génie civil.*

Présentation

Pierre HUMBERT
Chef de la Section Modèles numériques
Service Mécanique
Laboratoire Central des Ponts et Chaussées

Le progiciel CESAR-LCPC représente une très grande part de la compétence acquise par le Laboratoire Central des Ponts et Chaussées depuis une vingtaine d'années dans l'adaptation et l'application de la méthode des éléments finis aux problèmes de génie civil.

Historiquement, cette méthode a d'abord été développée par les ingénieurs de la construction mécanique (aéronautique, construction navale, ouvrages métalliques). Au LCPC, c'est sous l'impulsion initiale des géotechniciens, confrontés à des problèmes insolubles par les méthodes de calcul alors en vigueur, que la méthode a été introduite. À leur demande, un grand nombre de lois de comportement en mécanique des sols et des roches, ainsi que des procédés permettant la prise en compte du phasage de construction, ont été progressivement introduits dans le code de calcul. Les spécialistes des ouvrages d'art allaient rapidement se sentir concernés par la généralité de la méthode et provoquer des développements adaptés à leurs problèmes (éléments structuraux, modules de calcul en dynamique, etc.). Parallèlement, le développement de modules de calcul en diffusion transitoire a permis d'apporter une aide appréciable aux hydrogéologues et aux spécialistes de l'environnement. De même, dans le domaine des chaussées, des modules existants ou en cours de développement permettent d'aborder certains problèmes plus particuliers à ce type de structure (contact, propagation de fissure).

Ainsi, au fil du développement du progiciel, il apparaît que tous les domaines traditionnels des Laboratoires des Ponts et Chaussées se rejoignent sur une plate-forme de calcul commune.

Le développement, la diffusion et la maintenance d'une telle plate-forme nécessite l'existence d'une équipe ayant une bonne compétence dans les domaines évoqués ci-dessus. Cette équipe doit également avoir la maîtrise d'un certain nombre de techniques et de méthodes empruntées à des disciplines scientifiques (mathématiques, analyse numérique, mécanique des milieux continus, etc.), ou à des métiers (informaticien, dessinateur, graphiste, enseignant, etc.) très divers.

Les huitièmes journées d'informatique de Saint-Brieuc ont été l'occasion pour l'équipe de développement d'aborder à travers la Qualité un nouvel aspect méthodologique, et de mesurer l'impact de ce concept sur notre production. Cet effort nous permet de suivre avec attention la réflexion entreprise par le Service informatique pour doter le LCPC de véritables outils de génie logiciel.

Introduction

CESAR-LCPC est un progiciel* [1] de calcul basé sur la méthode des éléments finis, particulièrement adapté à la résolution des problèmes de génie civil et de génie industriel : calcul des structures, mécanique des sols et des roches, thermique, hydrogéologie, etc. [2]. Il se compose principalement du code de calcul CESAR et des pré- et post-processeurs interactifs graphiques MAX et PEGGY. On l'utilise dans de bonnes conditions sur les stations de travail actuelles ou sur tout autre calculateur scientifique ayant des caractéristiques analogues (mémoire centrale minimum de 8 Mo, espace disque de 100 Mo ou plus, écran graphique couleur haute définition).

CESAR-LCPC peut être considéré comme un gros progiciel dans la mesure où il est actuellement constitué d'un système de programmes dépassant les 150 000 instructions, de plus d'une centaine de jeux de données de test et d'une documentation répartie en une dizaine de types de manuels. Tout ceci n'a pu se développer sans un minimum de règles intuitivement mises en place à l'origine. Aujourd'hui, spécifications et méthodes sont clairement formalisées par le génie logiciel** [3].

A la méthodologie de développement établie dès l'origine du progiciel, sont venues s'ajouter au fil de sa croissance, des règles de gestion introduisant entre autres des critères de sûreté et de sécurité d'utilisation. CESAR-LCPC étant destiné à être diffusé, un ensemble de règles concernant la portabilité et la satisfaction des utilisateurs (ergonomie) est également présent, épousant parfois les contours de la qualité. En effet, « la qualité d'un logiciel n'est pas l'absence d'erreur dans son code (chose improuvable), mais plutôt son aptitude à satisfaire les besoins de l'utilisateur, tant sur le plan fonctionnel que du point de vue des caractéristiques d'utilisation » [4].

Rappel historique

L'actuel pôle de calcul est né à la fin des années soixante de la volonté de voir renforcés au LCPC les moyens d'études et de dimensionnement des ouvrages de génie civil et, en particulier, d'utiliser les éléments finis pour résoudre certains problèmes de géomécanique insolubles par les méthodes en vigueur à cette époque. C'est ainsi que dès 1968 était constituée une petite équipe pour développer un programme qui allait prendre le nom de ROSALIE. Les premières études réalisées avec cet outil allaient vite montrer l'intérêt et les potentialités qu'offrait la méthode. L'équipe de développement était alors renforcée pour lui permettre de construire un système complet, permettant le calcul des structures et de prendre en compte les comportements non linéaires tels que ceux rencontrés en mécanique et en hydrogéologie. Outre le programme de calcul proprement dit, ce système allait s'enrichir de programmes périphériques ayant pour objet la génération du maillage, la vérification des données du calcul et l'interprétation graphique des résultats [5].

L'architecture du système ROSALIE et la taille des problèmes traités nécessitaient l'emploi des gros calculateurs de l'époque (CDC 6600, IBM 370/168, etc.). Le réseau des LPC ne pouvait pas utiliser cet outil, car il n'était pas envisageable de l'installer sur les calculateurs alors disponibles dans les CETE. Aussi, à partir de 1975, la bibliothèque des PPR (Petits Programmes Rapides, de dimension restreinte et d'application spécialisée) était développée, avec comme principale spécification de pouvoir fonctionner sur des machines de la taille des IRIS 80, et de satisfaire ainsi la demande du réseau [6].

Entre l'origine de ROSALIE et le début des années quatre-vingt, le paysage informatique avait considérablement changé (disparition de la carte perforée, apparition de la mémoire vir-

---

* « Un progiciel comprend le dossier-programme et les supports d'information sur lesquels le programme est enregistré, les jeux d'essai et la documentation associée. Il constitue un ensemble cohérent et adaptable de programmes ou de sous-programmes permettant la réalisation par l'ordinateur d'une tâche importante et spécifique. »

** « Génie logiciel : ensemble des outils et de procédures relatifs aux différentes phases de l'élaboration d'un logiciel : spécification, programmation, suivi de production, test et mise au point, qualification, recette, maintenance, etc. »

tuelle, du temps partagé, télé-traitement, banalisation du graphique interactif), et avait fait naître de nouvelles exigences chez les utilisateurs : convivialité, facilité d'emploi, etc. De son côté, ROSALIE avait atteint un état critique à la fois en développement (taille et complexité du code), en maintenance (nombreuses redondances) et en utilisation (les programmes ne s'exécutaient qu'en traitement par lots). Il était donc devenu nécessaire de revoir nos méthodes de travail et de formuler de nouvelles spécifications pour le pôle « calcul par éléments finis ».

Genèse de CESAR-LCPC

Après analyse et critique du système ROSALIE et de la bibliothèque PPR, et à partir d'une étude bibliographique de la littérature « éléments finis » devenue abondante et désormais disponible en librairie, un certain nombre de spécifications étaient retenues, dont parmi les principales [7] :

— séparation des étapes « mise en données », « calcul », « exploitation des résultats » ;

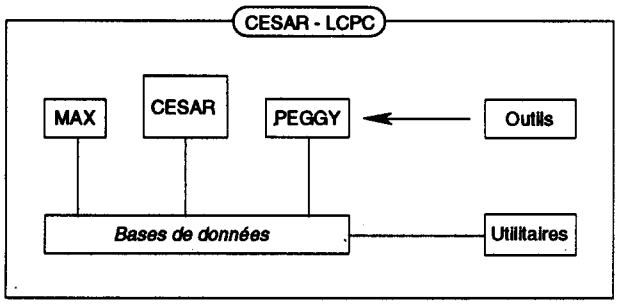
— progiciel de recherche, donc des programmes facilement développables : modularité ;

— progiciel industriel, donc des programmes facilement utilisables : interactif graphique ;

— rédaction d'une documentation complète et bien structurée ;

— diffusion sur une large gamme de matériel (micro-ordinateur exclu) : portabilité.

Le travail de programmation pouvait alors commencer pour aboutir actuellement à l'ensemble de programmes communiquant entre eux selon le schéma de la figure 1.



MAX : pré-processeur interactif graphique pour la réalisation du maillage et la préparation des données du calcul.

CESAR : code de calcul par éléments finis.

PEGGY : post-processeur interactif graphique pour l'interprétation des résultats du calcul et la création de documents.

Utilitaires : programmes d'aide à la mise en données (ils agissent sur la base de données).

Outils : programmes d'aide au développement (ils utilisent les fichiers source des programmes comme données)

Fig. 1 - Architecture du progiciel CESAR-LCPC.

Organisation du développement de CESAR-LCPC

L'équipe de développement

Dès l'origine, il était établi que le développement des programmes allait être assuré par une équipe spécialisée (actuellement répartie entre les différentes implantations géographiques du LCPC) à laquelle viendraient s'ajouter thésards et stagiaires pour la durée de leur recherche. Certains développements pourraient également être réalisés dans d'autres laboratoires (exemples : endommagement par l'École Normale Supérieure Cachan, « loi de Cambou » par l'École Centrale de Lyon, meilleur spécialiste en bio-mécanique par l'École Nationale Supérieure des Arts et Métiers, etc.). Compte tenu de la complexité croissante des algorithmes introduits dans les programmes, aucun développeur ne peut être un spécialiste de l'ensemble des domaines abordés par CESAR-LCPC. Cela a conduit à mettre l'accent sur deux aspects :

— nécessité d'une documentation complète, intégrant la phase de programmation, afin que plusieurs personnes puissent collaborer au développement sans risque de conflit ou de redondance ;

— architecture des programmes permettant de greffer les développements de manière simple, sans que chacun ait à connaître parfaitement l'ensemble du programme sur lequel il travaille.

La documentation

Un gros effort de documentation est demandé à l'équipe de développement car les manuels sont l'un des éléments visibles du progiciel (l'autre étant sa production graphique). Il faut donc souligner l'importance de celle-ci et y porter une attention au moins égale à celle de la programmation. Chacun doit considérer que la rédaction de la documentation fait partie du travail de développement : un minimum de rédaction s'impose de toute façon au programmeur, ne serait-ce que pour décrire l'ensemble des équations traitant le problème physique à modéliser (partie théorique), pour tracer l'organigramme du nouveau module à développer (architecture des instructions), pour décrire les données à fournir et les résultats obtenus (mode d'emploi), etc. Ainsi, loin d'apparaître comme une contrainte, la documentation des programmes devient une aide dans les phases de développement et de mise au point. En dernier lieu, et pour assurer le caractère industriel de CESAR-LCPC, les derniers développements ne peuvent être intégrés dans la version officielle que s'ils sont correctement et complètement documentés.

La documentation est organisée en différents manuels correspondant à un usage bien précis.

Pour conserver la modularité propre au progiciel et pour faciliter les mises à jour et les développements, la plupart des manuels sont organisés en fascicules indépendants les uns des autres. L'ordre dans lequel les différents manuels sont présentés ci-dessous correspond sensiblement à la chronologie que va suivre la rédaction de la documentation d'un nouvel algorithme introduit dans le progiciel :

— manuel théorique ;
— manuel de programmation ;
— mode d'emploi ;
— bibliothèque des jeux de données (dont recueil de solutions analytiques) ;
— bibliothèque d'exemples ;
— manuel de formation (dont fascicule de présentation, glossaire, etc.) ;
— fascicule d'implantation ;
— recueil d'incidents.

Le manuel de programmation

Le manuel de programmation retient ici notre attention dans la mesure où c'est à partir de lui que la notion de qualité va pénétrer les différents constituants du progiciel. Ce manuel contient en effet les règles d'architecture, de programmation et de documentation. Il présente les organigrammes et leur traduction en langage de programmation (fichiers source), il décrit également les différents fichiers de la base de données assurant la communication entre les programmes et il expose les règles de gestion des différentes versions. Un fascicule particulier est chargé d'afficher les axes de développement futurs. L'objectif du manuel de programmation est en premier lieu d'assurer la maintenance et de faciliter la correction d'éventuelles erreurs. En second lieu, il permet aux programmeurs chargés des développements d'effectuer ceux-ci en harmonie avec les algorithmes déjà codés.

Règles d'architecture : exemple du code de calcul CESAR

CESAR est constitué de modules de gestion des données (exemple : COOR pour la définition des coordonnées des nœuds, CHAR pour la définition des chargements, etc.), de modules d'exécution (exemple : LINE pour la résolution d'un problème linéaire, MCNL pour un problème de mécanique à comportement non linéaire, etc.), de familles d'éléments (exemple : famille 1 regroupant les éléments bidimensionnels en mécanique, famille 5 pour les éléments de coque, famille 22 pour les éléments tridimensionnels en diffusion, etc.) d'une interface réalisant l'aiguillage entre modules et les familles d'éléments concernées, et d'une librairie de fonctions générales.

La figure 2 fait clairement apparaître la modularité de CESAR et l'indépendance qui existe d'une part entre l'entrée des données et le traitement, et d'autre part entre la programmation des familles et celle des modules d'exécution.

La sortie des résultats (impressions, alimentation de la base de données) est réalisée dans des unités de programmes spécifiques, indépendantes de celles chargées du traitement proprement dit. Elles n'apparaissent pas sur la figure 2 car étant très dépendantes de la famille d'éléments ou du module d'exécution utilisé, il est plus commode de les rattacher à ceux-ci.

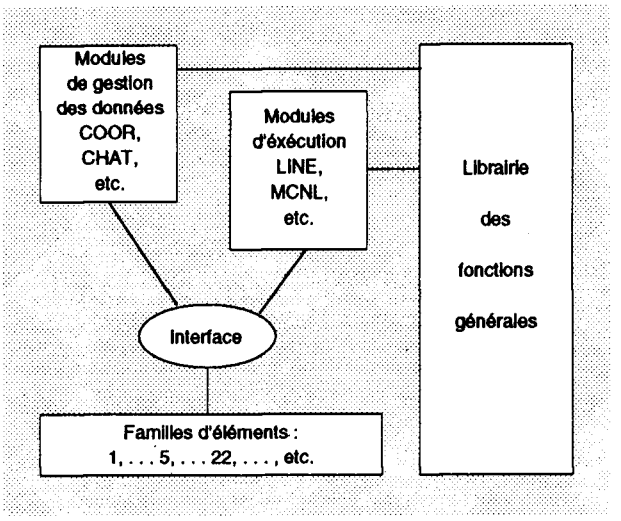

Modules de gestion des données COOR, CHAT, etc.
Modules d'éxécution LINE, MCNL, etc.
Librairie des fonctions générales
Interface
Familles d'éléments : 1, . . . 5, . . . 22, . . . , etc.

Fig. 2 - Architecture du code de calcul CESAR.

La gestion de l'espace réservé pour les tableaux est un autre point important de l'architecture du code de calcul. Schématiquement, CESAR utilise une table unique et des pointeurs permettant de découper celle-ci en différents tableaux créés par l'emploi des modules activés par l'utilisateur. Ceux-ci sont alloués en premier, la place non utilisée étant ensuite attribuée à la matrice de rigidité. Si cette dernière sort de la place disponible, CESAR déclenche alors un mécanisme de pagination sur disque. La gestion des pointeurs ainsi que la réservation, la restitution et la compression d'espace, sont assurées à l'intérieur des modules par appel des fonctions de la librairie générale. Cette technique d'allocation pseudo-dynamique permet de régler facilement le programme à la taille des problèmes usuellement traités et d'optimiser ainsi l'emploi des ressources informatiques (adéquation mémoire centrale / mémoire de masse / temps de calcul).

Le manuel de programmation présente également les règles pour insérer un nouveau module ou une nouvelle famille d'éléments et les modifications que cela implique dans certaines unités de programme de l'interface.

Règles de programmation

L'écriture des unités de programmes doit évidemment se faire en respectant un certain nombre de règles ayant trait à la lisibilité (l), à la portabilité (p), etc. On peut citer :

— les règles de rangement des unités de programmes dans les fichiers source (l) ;
— l'ordre des instructions de déclaration (l) ;

— les règles concernant les formats d'impression (*p*) ;
— l'utilisation de la programmation structurée ;
— les règles d'attribution des noms de variables et d'unités de programmes ;
— les règles d'utilisation des fonctions intrinsèques du langage de programmation (*p*).

Les programmes interactifs graphiques : MAX et PEGGY

Les fascicules consacrés à MAX et à PEGGY contiennent les informations indispensables pour conserver l'homogénéité d'apparence des divers modules graphiques. On trouvera donc la description des différentes fenêtres écran et traceur, les caractéristiques de couleur, taille, épaisseur, etc., retenues pour les primitives graphiques (ligne, texte, symbole, remplissage de zone) ainsi que l'environnement des interactions utilisées (saisie clavier/réticule puis décodage).

Règles de documentation

Les règles et les conseils concernant la rédaction des manuels, déjà évoqués précédemment, figurent également dans le manuel de programmation. Un complément indispensable à ce dernier consiste à documenter les fichiers source à l'aide de commentaires. En particulier, en tête de chaque unité de programme on doit trouver :

— un commentaire décrivant la fonction de l'unité de programme ;

— des commentaires relatifs à la liste des arguments d'appel à l'unité de programme identifiant les arguments servant de données et ceux contenant les résultats des calculs effectués ;

— des commentaires identifiant les arguments contenus dans les zones communes et modifiés dans l'unité de programme ;

— la liste des fonctions externes à la librairie des fonctions intrinsèques et qui risquent donc de créer des problèmes de portabilité (par exemple, tirage de nombres aléatoires, mesure du temps de calcul, produit scalaire, etc.) ;

— la liste des fichiers manipulés ;

— des remarques éventuelles sur les limitations de l'algorithmique actuelle, les difficultés d'emploi ou de portabilité, et les améliorations à envisager ;

— des commentaires identifiant les corrections ou modifications apportées à la version originale avec leurs dates.

Les versions de programmes

Valoriser les développements effectués pour la recherche en les mettant dès que possible à la disposition des clients du LCPC (réseau LPC, milieu industriel, milieu de recherche/enseignement) est une de nos ambitions. CESAR-LCPC doit donc être à la fois un progiciel industriel et un progiciel de recherche.

Un progiciel industriel est un produit fini constitué de programmes entièrement validés, accompagnés d'une documentation complète et des jeux de données de test. La sécurité des algorithmes présents dans le progiciel doit être telle que, après un minimum de formation, nos « clients » soient en mesure d'utiliser CESAR-LCPC de façon autonome sur leur propre calculateur. Sous réserve d'une modélisation correcte du problème traité, la fiabilité des résultats doit leur être garantie.

Un progiciel de recherche offre un cadre moins rigide que celui décrit ci-dessus. Il s'agit plutôt d'un produit en cours d'élaboration fonctionnant comme une structure d'accueil sur laquelle de nombreux développeurs peuvent intervenir. Les objectifs sont alors multiples et les interventions sont orientées principalement dans les directions suivantes :

— développements finalisés : il s'agit d'introduire les algorithmes issus du fascicule « Axes de développement » qui se retrouveront dans la version "industrielle" ;

— développements prospectifs : il s'agit de tester de nouveaux algorithmes. Après analyse et critique, ils peuvent ne pas être retenus pour la version « industrielle » (exemples : problème de convergence ou de temps de calcul, ou, encore, contrat de développement soumis à confidentialité) ou laissés en veille technologique (exemple : emploi trop délicat) ;

— « taillage » du code pour modéliser un problème particulier ou pour l'adapter à une machine particulière (exemples : version réduite pour essais de machines, implantation sur machines « exotiques », etc.).

Pour répondre à ces spécifications, chaque nouvelle fonctionnalité va passer par une étape de développement, puis par une étape d'intégration dans le progiciel pour atteindre l'étape finale de la diffusion aux "clients". Il est évident que CESAR-LCPC est, et sera toujours, en constante évolution et que ce processus est continu dans le temps. Tout utilisateur doit pouvoir clairement identifier la version utilisée ainsi que la provenance des résultats obtenus. C'est pourquoi les étapes décrites ci-dessus sont respectivement nommées « phase recherche », « phase beta-test » et « phase standard » et que chaque programme édite ce nom et le numéro de version à tous les endroits stratégiques :

— dans les dialogues à l'écran ;
— dans les fichiers d'édition ;
— dans les fichiers binaires (surtout s'il y a des risques d'incompatibilité entre versions) ;
— dans les fichiers destinés aux traceurs.

Les outils

Pour pallier la carence de certains systèmes d'exploitation utilisés au cours du développement de MAX, de CESAR et de PEGGY, des programmes ont été écrits pour venir en aide aux développeurs. Actuellement, des fonctions telles que organigramme, édition de fichiers source, index des unités de programmes, manipulation/comparaison de fichiers symboliques, analyse des zones de variables communes, etc., peuvent être réalisées par ces petits programmes qui forment la « boîte à outils » de CESAR-LCPC. La durée de vie de ces outils est évidemment très dépendante de l'introduction de véritables outils de génie logiciel sur le réseau Unix du LCPC.

Validation

D'une façon générale, il est établi que la validation d'un logiciel scientifique comporte deux aspects [8] :

« — la vérification des modules élémentaires qui réalisent une fonction mathématique ou un algorithme numérique précis. Elle se fait par autocontrôle au niveau du réalisateur ;

— la validation globale du logiciel qui intègre ces modules vérifiés. Elle nécessite une comparaison avec des résultats théoriques, expérimentaux, ainsi qu'avec d'éventuels résultats obtenus par des logiciels utilisant d'autres algorithmes de résolution. Car ce n'est pas le logiciel proprement dit qui a besoin d'être validé, c'est en fait l'ensemble des applications qu'il permet de résoudre, avec pour chacune d'elles une plage de validité précise. »

Pour CESAR-LCPC, nous distinguons le cas des pré- et post-processeurs pour lesquels une vérification visuelle suffit pour valider les différents modules graphiques, du cas du code de calcul, pour lequel nous appliquons des règles voisines de celles qui viennent d'être énoncées :

a) Recherche de cas test ayant une solution analytique et résolution par éléments finis. La comparaison entre ces résultats analytiques et numériques fonde la validation de la programmation sur ces cas particuliers [9].

b) Calage des paramètres mécaniques des lois de comportement en modélisant par éléments finis les essais de laboratoire (validation du progiciel, mais aussi validation de la loi de comportement choisie).

c) Validation du progiciel par comparaison avec les résultats expérimentaux provenant d'essais *in situ,* effectués sur ouvrages réels.

Les jeux de données utilisés et les résultats obtenus pour a) et b) sont en général répertoriés dans le manuel « Bibliothèque des jeux de données ». La méthodologie complète mise en œuvre pour c) est plutôt décrite dans les fascicules de la « Bibliothèque d'exemples ».

Utilisation et diffusion

Le progiciel CESAR-LCPC est utilisé en interne pour de nombreux travaux de recherche et dans des interventions d'experts. Il est également diffusé dans le réseau des LPC, dans le réseau technique du Ministère, en milieu industriel et dans le milieu recherche-enseignement. Une soixantaine d'implantations ont été réalisées sous divers systèmes d'exploitation, le standard recommandé étant Unix. La diffusion sur des machines très diverses est rendue possible grâce à l'application de règles exposées ci-dessous, et présentes bien sûr dans le manuel de programmation.

Langage et bibliothèque graphique

L'écriture des programmes se fait avec le langage Fortran 77, en excluant toutes les extensions à la norme que pourraient offrir les compilateurs utilisés.

La bibliothèque graphique utilisée par les pré- et post-processeurs est GKS. Cette bibliothèque autorise un niveau d'interactivité satisfaisant et permet surtout d'être indépendant des unités graphiques (écrans, traceurs, imprimantes, etc.) sur lesquelles les programmes vont dessiner.

Portabilité

La portabilité ne se décrète pas *a priori,* mais se constate lors d'une implantation sur une configuration (machine, système d'exploitation, compilateur, implémentation GKS) que l'on n'avait pas encore rencontrée. Certains conseils peuvent néanmoins être précieux [10], associés au strict respect des normes Fortran 77 et GKS. Malgré ces précautions, certaines instructions posent des problèmes de portabilité : on peut citer les instructions relatives à la gestion des fichiers, l'acquisition de la date, de l'heure, la mesure du temps de calcul, l'emploi de dispositifs particuliers améliorant les performances (typiquement, calcul du produit scalaire), les initialisations de variables dépendant de l'implémentation GKS (exemple : numéros des polices de caractères, des symboles, des pilotes d'unités graphiques, etc.). Pour pouvoir réaliser facilement de nouvelles implantations, les instructions dépendant de la machine et du système d'exploitation, de l'implémentation GKS et des terminaux graphiques ont été regroupés dans des modules particuliers.

CESAR-LCPC et la qualité

Pour suivre les recommandations de l'Association Française pour le Contrôle Industriel et la Qualité (AFCIQ), l'évaluation de la qualité devrait être confiée à une « cellule qualité », intervenant, par exemple, sur toute la production informatique du LCPC (voire du réseau) destinée à être diffusé. Anticipant sur la mise

en place d'un plan qualité logiciel au sein des LPC, la section Modèles numériques se risque donc à évaluer un progiciel dont elle est le principal auteur. Toutefois, et compte tenu des effectifs que le LCPC peut raisonnablement mobiliser autour de CESAR-LCPC, ne perdons pas de vue que « la qualité est un compromis entre ce qui est souhaitable et ce qui est possible... » [4].

Aspect qualité

Au cours des paragraphes précédents, le lecteur aura pu identifier certains critères qui procèdent de la qualité :

— richesse des fonctionnalités (voir [2]) ;
— facilité d'emploi grâce aux pré- et post-processeurs interactifs graphiques ;
— documentation bien structurée ;
— modularité facilitant le développement, la portabilité et la maintenance ;
— fiabilité : aspect validation ;
— gestion de différentes versions ;
— portage sur de nombreuses machines en particulier grâce au langage Fortran ;
— indépendance vis-à-vis des postes de travail graphiques : GKS ;
— développements pouvant être répartis entre plusieurs équipes.

La production d'un logiciel de qualité consiste également à passer par les phases classiques de développement. On peut ainsi mettre en face de chaque phase un document d'orientation ou un manuel du progiciel CESAR-LCPC :

0 - Expression des besoins	→ Recherche programmée, sujet de thèses ou fascicule « Axes de développement »
1 - Spécifications	→ Compte-rendu de réunion avec le chef de projet
2 - Conception	→ « Manuel Théorique », et fascicule « Description du programme xxx »
3 - Réalisation/ codage	→ Fichiers source du programme xxx
4 - Validation/ tests	→ « Bibliothèque des jeux de données de test »
5 - Exploitation/ maintenance	→ « Recueil d'incidents »

Aspect non-qualité

L'équipe de développement identifie dans sa production actuelle certains aspects de non-qualité :

• Fortran 77 n'est pas un langage garantissant une programmation de qualité et mettant à l'abri des effets de bord, des débordements de tableaux ou des débranchements intempestifs. Il exige donc une grande vigilance lors de la phase d'intégration des développements.

• Bien que nous y portions une attention particulière, l'ergonomie des pré- et post-processeurs n'atteint pas celle qu'un utilisateur peut trouver avec les logiciels industriels de la micro-informatique ou de la CAO (Conception Assistée par Ordinateur).

• Certaines redondances sont apparues entre les programmes 2D et 3D des pré- et post-processeurs de la version officielle à cause notamment de leur développement à un moment donné sur des machines non compatibles (Sintran et Unix) et surtout non réseautées.

• La procédure de correction d'erreurs, héritée des habitudes passées, nécessite beaucoup trop d'interventions humaines.

• Il n'y a pas de séparation entre équipe de développement et équipe de diffusion/maintenance.

Amélioration de la qualité

Dans le but d'améliorer la qualité, certaines modifications ont déjà été apportées à la version actuellement en fin de validation, comme, par exemple, la création de modules communs pour supprimer les redondances apparues dans MAX et PEGGY. Notre méthodologie des corrections doit également être repensée, en utilisant par exemple sous Unix l'outil standard SCCS (Source Code Control System).

La forte image de marque « génie civil » de CESAR-LCPC vient naturellement des possibilités de modélisation qu'offre le code de calcul par éléments finis CESAR (familles d'éléments, modules d'exécution, lois de comportement, etc.) et c'est plutôt ce domaine qui constitue le « métier » de l'équipe de développement. Néanmoins, nous avons conscience que CESAR est de plus en plus utilisé comme une boîte noire et que c'est sur les pré- et post-processeurs interactifs graphiques MAX et PEGGY que l'ensemble du progiciel est jugé. Il est ainsi assuré que les utilisateurs seront sensibles à la recherche d'une meilleure ergonomie, ainsi qu'à l'interfaçage avec les produits standards de la CAO.

CESAR-LCPC étant utilisé en France par des non francophones et à l'étranger (Allemagne, Grèce, Portugal, Brésil, Suède, Canada), nous réfléchissons à une organisation des programmes permettant le multilinguisme. Pour la documentation, il semble raisonnable de s'en tenir à l'anglais.

On pourrait également discuter le choix du langage de programmation puisque celui-ci apparaît à la fois comme critère de qualité et de non-qualité. Compte tenu du nombre d'instructions déjà rédigées et de l'aspect diffusion sur un grand nombre de machines, CESAR-LCPC est, et restera, écrit en Fortran. Ce point pourra être réexaminé lorsqu'il deviendra nécessaire de donner un successeur à CESAR-LCPC... Mais peut-être y a-t-il beaucoup à attendre du nouveau langage Fortran 90.

Conclusion

Après l'expérience du système ROSALIE et de la bibliothèque des PPR, nous avions créé notre propre échelle de valeurs pour ranger les spécifications et les critères de qualité de notre actuel pôle de calcul. Cette notion de qualité est maintenant formalisée par les travaux de l'AFCIQ et de l'AFNOR. Si elle devient une exigence chez un nombre croissant d'utilisateurs et de destinataires des résultats de calcul, elle doit surtout être prise en compte à chaque phase de développement. La qualité doit s'appliquer autant aux logiciels destinés à être diffusés qu'à ceux soumis à une utilisation interne intensive, et devenir une grille de lecture de la production informatique de chacun.

CESAR-LCPC ayant été construit sur des bases saines, une première évaluation de ses aspects qualité a rassuré l'équipe de développement qui aurait pu voir la qualité comme une nouvelle mode et comme une contrainte ralentissant et bridant les développements futurs. Pouvant maintenant nous appuyer sur la réflexion entreprise dans le réseau des LPC depuis les journées d'informatique de Saint-Brieuc [11], l'aspect qualité doit être clairement affiché et, en particulier, pris en compte dans le Manuel de programmation de CESAR-LCPC. Nous veillerons toutefois à limiter le nombre de critères et de spécifications pour éviter tout phénomène de rejet de la part des développeurs, ce qui irait à l'encontre de la notion même de qualité.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] IBM Corporation (1980), *Terminologie du traitement de l'information,* Publication IBM, 340 p.

[2] HUMBERT P. (1989), CESAR-LCPC : Un code de calcul par éléments finis, *Bull. liaison Labo. P. et Ch.*, **160,** mars-avril, pp. 112-115.

[3] 01-Référence (1989), *Les mots de l'année,* Périodique 01-Informatique, Hors série n°3.

[4] NEEL D. (1987), *La notion de qualité dans le cas des logiciels,* Note CNRS, Direction de la Valorisation et des Applications de la Recherche.

[5] GUELLEC P. (1976), ROSALIE : système de calcul des massifs et des structures. Notice de présentation et d'utilisation en géomécanique et en mécanique des structures, Publication LCPC, Section Modèles numériques, (épuisé).

[6] HUMBERT P. (1981), *Les systèmes de calcul par éléments finis du LCPC,* Compte-rendu des Journées de Physique, Les Arcs.

[7] FEZANS G. (1983), *Présentation du code* CESAR *: architecture et possibilités actuelles et futures,* Séminaires Recherche du LCPC. Exposés - Débats, (vidéo-cassette de la séance du 12 janvier 1984), Service audiovisuel du LCPC.

[8] NEEL D. (1987), *Recommandations pour l'assurance et le contrôle qualité des logiciels scientifiques (Manuel qualité),* Note CNRS, Direction de la Valorisation et des Applications de la Recherche.

[9] MESTAT P. (1987), *Recueil de solutions analytiques pour validation des codes de calcul aux éléments finis,* Publication LCPC, Section des Fondations et soutènements (FAER 1.05.10.7).

[10] FICHEUX-VAPNE F. et *al.* (1985), Fortran 77 : *Guide pour l'écriture de programmes portables,* Collection DER-EDF, Ed. Eyrolles, 150 p.

[11] CHERVET G. et *al.* (1991), *Propositions pour un Manuel Qualité des applications informatiques des LPC,* doc. du LRPC de Saint-Brieuc.